UMIACS-TR-90-6
CS-TR-2388

January 1990

The Definition of Necessary Hidden Units in Neural Networks for Combinatorial Optimization

Benjamin J. Hellstrom and Laveen N. Kanal†
Institute for Advanced Computer Studies,
Machine Intelligence and Pattern Analysis Laboratory and
Department of Computer Science
University of Maryland
College Park, MD 20742

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

UMIACS–TR–90–6 January 1990
CS–TR–2388

The Definition of Necessary Hidden Units in Neural Networks for Combinatorial Optimization

Benjamin J. Hellstrom and Laveen N. Kanal[†]
Institute for Advanced Computer Studies,
Machine Intelligence and Pattern Analysis Laboratory and
Department of Computer Science
University of Maryland
College Park, MD 20742

ABSTRACT

Among the earliest successful applications of multi-layered neural networks are combinatorial optimization problems, most notably, the travelling salesman problem. Hopfield-type thermodynamic networks comprised of functionally homogenous *visible* units have been applied to a variety of structurally simple *NP*-hard optimization problems. A fundamental obstacle to the application of neural networks to difficult problems is that these problems must first be reduced to 0-1 Hamiltonian optimization problems. We show that certain optimization problems cannot be embedded in networks composed entirely of visible units and present a method for defining necessary *hidden units* together with their *best features*. We derive a knapsack-packing network of *O(n)* units with both standard and conjuntive synapses. Encouraging simulation results are cited.

(KR)

[†] Research supported in part by NSF grant DCR8504011.

Introduction

Since their resurgence in the early 1980s artificial neural systems have found applications in computer vision, speech generation and recognition, robotics, and numerous other areas. Progress toward the application of neural networks to *NP*-hard combinatorial optimization problems has been modest and has been generally restricted to structurally simple problems. The first and most well-known application was presented by Hopfield and Tank [Hop85] who reduced the travelling salesman optimization problem (TSP) to a 0-1 quadratic assignment optimization problem (QAP) with a Hamiltonian objective function. Hopfield and Tank then showed that a thermodynamic neural network with symmetric connections and a non-linear sigmoid transfer function could effectively find good solutions to embedded TSP problems. The approach to reducing optimization problems that was proposed by Hopfield and Tank has since been applied to numerous other combinatorial optimization problems. J. Ramanujam and P. Sadayappan present reductions of graph partitioning, graph K-partitioning, minimum vertex cover, maximum independent set, maximum clique, set partition, and maximum matching to QAPs [Ram88]. E.D. Dahl presents reductions of map and graph coloring problems [Dah88].

A common characteristic of all these problems is that they are structurally simple and can easily be reduced to QAPs with Hamiltonian objective functions. Furthermore, all of the resulting neural networks consist of functionally homogeneous processing units whose activation values are directly mapped to the solutions of their respective embedded optimization problems. Since these units participate in the expression of problem solutions for external interpretation, they can be viewed as *visible units*.

The Set Partition Problem

Before we consider a difficult optimization problem, we review Hopfield and Tank's reduction technique on a simple problem. The integer set partition optimization problem is given by

INSTANCE: Finite set A of elements, for each $a \in A$, a size $b_a \in Z^+$.

OBJECTIVE: Give a subset $A' \subseteq A$ such that

$$\left| \sum_{a \in A'} b_a - \sum_{a \in A-A'} b_a \right|$$

is minimized over all subsets of A.

The set partition decision problem is known to be *NP*-complete [Gar79].

Let **V** be the variable space of the set partition problem. **V** is the set of all subsets of A so that $|\mathbf{V}|=2^{|A|}$. Let **S** be the space defined by the volume of an n-dimensional unit hypercube for some, as of yet unspecified n. Each point in **V** or configuration of problem variables is associated with a configuration of the states of n neuron-like units, that is, a point in **S**. This association is defined by a pair of mappings M:**V**→**S**, and M^{-1}:**S**→**V**. We could define $n = 2^{|A|}$ and map each subset of A to a unique vector in the set of n orthogonal, unit-length, binary vectors. However, this results in a neural network whose size scales exponentially with the size of the problem.

A better approach is to define $n = |A|$ and to map each subset of A to a unique combination of n binary values. Let us name the elements of A by $\{ a_1, a_2, \ldots, a_n \}$. We define M, for the subset of **S** consisting of the corners of the n-dimensional hypercube, by

$$M(A) = (s_1, s_2, \ldots, s_n)$$

where

$$s_i = \left[\begin{array}{l} 1 \ if \ a_i \in A' \\ \\ 0 \ if \ a_i \in A - A' \end{array} \right]$$

$$A' \subseteq A, \ 1 \leq i \leq n.$$

In order to extract a useful problem solution from a network configuration, we must also define M^{-1}. This is typically done via application of a threshold, χ :

$$M^{-1}(\ s_1,\ s_2,\ \ldots,\ s_n\ ) = \{\ a_i \mid s_i \geq \chi\ \}$$

$$0 < \chi < 1,\ 1 \leq i \leq n.$$

This completes our selection of a representation, that is, the definition of n and the mapping of problem variable configurations to, and from the set of global network configurations.

The next step in the derivation process is the selection of an appropriate *energy function* $E : \mathbf{S} \rightarrow \Re$. Hopfield and Tank showed that in order to embed the problem in a network of neuron-like processing units, the energy function must be expressible as a *0-1 Hamiltonian*. Hamiltonian energy functions have the form -

$$E = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} s_i s_j w_{ij} + \sum_{i=1}^{n} s_i \theta_i$$

where n is the number of processing units in the network, s_i is the activation level of the i^{th} unit, w_{ij} is the connection strength between the i^{th} and j^{th} units, and θ_i is the activation threshold of the i^{th} unit. All connections are symmetric, i.e., $w_{ij} = w_{ji}$ for all i, j. The fundamental obstacles to embedding arbitrary optimization problems in neural networks are the discovery of a representation and a Hamiltonian energy function so that minimal-energy network configurations are mapped to optimal configurations of problem variables. For the set partition problem, we choose the energy function -

$$E = B \cdot \left(\sum_{i=1}^{n} s_i b_i - \sum_{i=1}^{n} (1-s_i) b_i \right)^2$$

It can easily be seen that $\sum_{i=1}^{n} s_i b_i$ is the sum of sizes of the elements in A' and $\sum_{i=1}^{n} (1-s_i) b_i$ is the sum of sizes of elements in $A-A'$. Without some insight into the forms of valid energy functions, we must manipulate E algebraically before we can be certain that it can be expressed as a 0-1 Hamiltonian.

$$E = B \cdot \left(\sum_{i=1}^{n} s_i b_i - \sum_{i=1}^{n} (1-s_i) b_i \right)^2$$

$$= B \cdot \left(2 \sum_{i=1}^{n} s_i b_i - \sum_{i=1}^{n} b_i \right)^2$$

$$= B \cdot \left(4 \sum_{i=1}^{n} \sum_{j=1}^{n} s_i s_j b_i b_j - 4 \sum_{i=1}^{n} s_i b_i \left(\sum_{j=1}^{n} b_j \right) + \left(\sum_{j=1}^{n} b_j \right)^2 \right)$$

Since $\left(\sum_{j=1}^{n} b_j \right)^2$ is constant with respect to s_i we can drop it from E without affecting the minima. The constant coefficients (4) can be included in B and are dropped as well. We are left with

$$E = B \cdot \sum_{i=1}^{n} \sum_{j=1}^{n} s_i s_j b_i b_j - B \cdot \sum_{i=1}^{n} s_i b_i \left(\sum_{j=1}^{n} b_j \right)$$

$$= -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} s_i s_j (-2B\, b_i b_j) + \sum_{i=1}^{n} s_i \left(-B\, b_i \sum_{j=1}^{n} b_j \right)$$

This expression is in the standard form of a 0-1 Hamiltonian where

$$w_{ij} = -2B\, b_i b_j$$

and

$$\theta_i = -B\, b_i \sum_{j=1}^{n} b_j \, .$$

The derivative of E allows us to deduce the function of unit i -

$$\frac{\partial E}{\partial s_i} = -\sum_{j=1}^{n} s_j w_{ij} + \theta_i$$

$$= 2Bb_i \left(\sum_{j=1}^{n} s_j b_j - \frac{1}{2} \sum_{j=1}^{n} b_j \right).$$

In order to minimize E, the value of s_i should increase whenever $\frac{\partial E}{\partial s_i} < 0$. This corresponds to the condition

$$\sum_{j=1}^{n} s_j b_j < \frac{1}{2} \sum_{j=1}^{n} b_j \, .$$

Each unit behaves as a feature detector, detecting the condition in which the sum of the sizes of elements in A' is less than half of the total sum of sizes. By detecting this feature, unit i increases its level of activation and gradually moves its associated element a_i from $A-A'$ into A'. The result is a decrease the discrepancy between the sums of sizes of elements in the two sets.

The local function of unit i $(1 \leq i \leq n)$ in a Hopfield-type network is given by

```
repeat
    $\Delta E \leftarrow -\sum_{j=1}^{n} s_j w_{ij} + \theta_i$ ;
    $s_i \leftarrow \tau \cdot s_i + (1-\tau) \cdot \left( \frac{1}{1 + e^{(-\Delta E / T)}} \right)$ ;
until (externally terminated);
```

Where τ $(0 < \tau \leq 1)$ controls the response time of the unit. As $\tau \downarrow 0$, the trajectory of the network configuration becomes increasingly smooth. Units can be updated either synchronously, as prescribed by Hopfield and Tank, or asynchronously. After relaxation, the approximate solution to the embedded set partition problem is extracted from the configuration of activation values by application of M^{-1}.

It should be noted that **V** is often a discrete space while **S** is continuous. The set of minima of a continuous

0-1 Hamiltonian must be a subset of the set of corners of the n-dimensional unit hypercube. There are few restrictions on M except those imposed by the definitions of its domain and range. If there is *no* prior knowledge about the minima of f, then we must take care to ensure that *all* points in $\mathbf{V}$ are mapped by M to corners of the hypercube. Otherwise, we can not be certain that the minima of f map to minima of E.

Let us summarize the basic techniques that are employed to find solutions to optimization problems using neural networks

1 A network representation is selected. n, the dimensionality of the network state space (alternatively, the number of units in the network architecture) is defined. A transformation, M:$\mathbf{V}\rightarrow\mathbf{S}$, from $\mathbf{V}$, the space of problem variable configurations, to $\mathbf{S} = [0,1]^n$, the volume of an n-dimensional unit hypercube, is defined.

2 An inverse transformation, M^{-1}:$\mathbf{S}\rightarrow\mathbf{V}$, is defined. Each network configuration in the volume of the n-dimensional hypercube is mapped by M^{-1} to a configuration of the variables of the problem space.

3 An energy function, $E:\mathbf{S}\rightarrow\Re$, is defined on $\mathbf{S}$ so that $M^{-1}(\mathbf{s}_{\mathrm{min}})$ is a minimum of the problem objective, f, whenever $\mathbf{s}_{\mathrm{min}}$, is a minimum of E, $\mathbf{s}_{\mathrm{min}}\in\mathbf{S}$. In order to utilize processing units with neuron-like behaviors, E must take the form of a 0-1 Hamiltonian.

4 The network is relaxed using one of a variety of unit update rules. Prior to relaxation, all network parameters (e.g. B in the set partition problem and operational parameters (e.g. τ) are specified.

5 After relaxation, M^{-1} is applied to the final network configuration yielding a minimum of f, the objective function of the original optimization problem.

Steps 1-3 are, without question, the most difficult and require a degree of cleverness to carry out. The technique is summarized in figure 1.

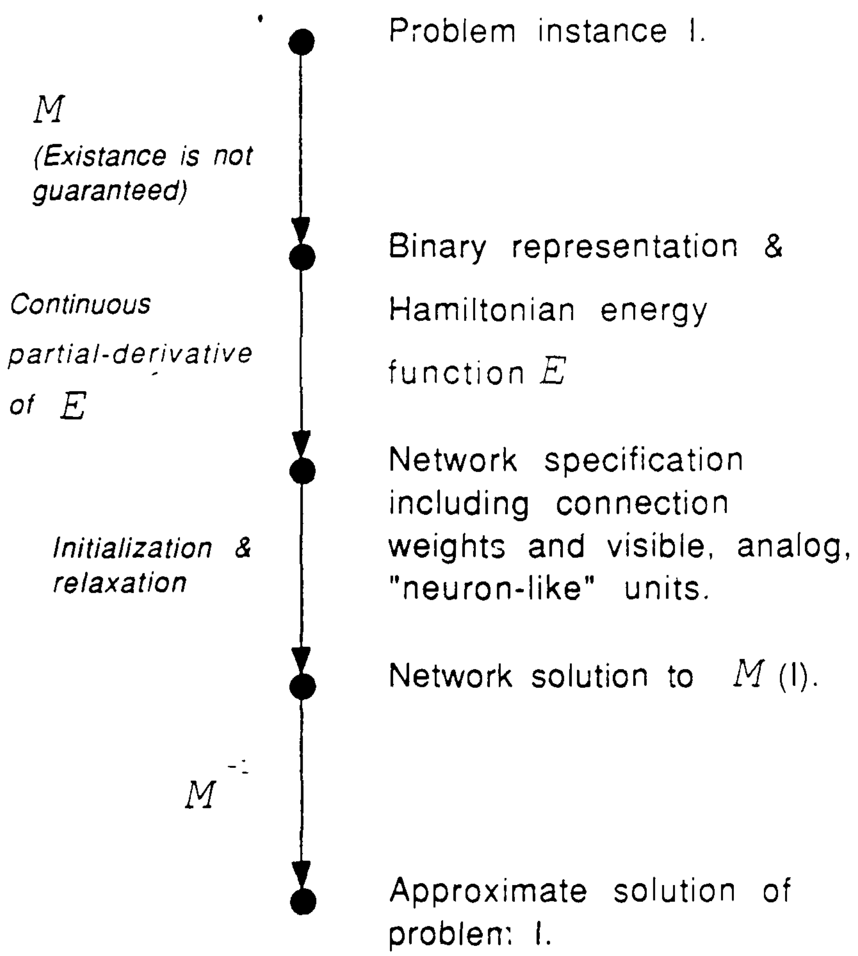

Problem instance I.
M
(Existance is not guaranteed)
Binary representation & Hamiltonian energy function E
Continuous partial-derivative of E
Network specification including connection weights and visible, analog, "neuron-like" units.
Initialization & relaxation
Network solution to M (I).
M^{-1}
Approximate solution of problem I.

Figure 1

Existing Thermodynamic Neural Network Derivation Procedure

In some sense, we have formalized the reduction of discrete minimization problems to the minimization of continuous 0-1 Hamiltonians. In subsequent discussions we will, on occasion, refer to these formal concepts but will not burden ourselves with strict adherence to formality.

The Knapsack Problem

Let us consider the integer knapsack problem given by

INSTANCE: Finite set $Q = \{1,2,...,n\}$ of elements, for each $q \in Q$ a cost $w_q \in Z^+$ and a profit $p_q \in Z^+$ and a positive integer knapsack capacity, K.

OBJECTIVE: Give a subset $Q' \subseteq Q$ such that

$\sum_{q \in Q'} w_q \leq K$ and $\sum_{q \in Q'} p_q$ is maximized over all subsets of Q.

As with the set partition decision problem, the integer knapsack decision problem is known to be *NP*-complete [Gar79]. The standard approach for embedding this problem in a neural network calls for the mapping of problem variable configurations, i.e. subsets of Q, to configurations of a set of n 0-1 variables representing the activation values, s_i, $1 \leq i \leq n$, of neuron-like processing units. The simplest representation is

$$s_i = \begin{bmatrix} 1 & if\ i \in Q' \\ 0 & otherwise \end{bmatrix}$$

where $n = |Q|$. We propose a global energy function, $E = E_A + E_B$ where E_A is a term reflecting the benefi. of maximizing $\sum_{q \in Q'} p_q$ and E_B is a term reflecting the penalty for violating

$$\sum_{q \in Q'} w_q \leq K.$$

What features should be detected by unit i $(1 \leq i \leq n)$? Since $s_i = 1$ whenever element i is placed in the knapsack, s_i should attain this value whenever the benefit (with respect to minimizing E) of placing element i in the knapsack outweighs the penalty. In other words $\Delta s_i > 0$ whenever $\frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i} < 0$. If $\frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i}$ can be computed as the sum of weighted activation values of s_j $(j \neq i)$, then the function of unit i will be to detect the condition

$$\mathbf{C1} \equiv \left(\frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i} < 0 \right).$$

When $\Delta s_i = 1$, ΔE_A should be proportional to $-p_i$ because the inclusion of element i in the knapsack increases the profit of the packing by effecting a decrease in E. The energy penalty, ΔE_B, depends on the states of the other units in the network.

Let

$$\mathbf{C2} \equiv \left(K < \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \right).$$

If **C2** is true then ΔE_B should be proportional to w_i. If **C2** is "strongly" false then there should be no energy penalty, that is, $\Delta E_B = 0$. $\frac{\Delta E_B}{\Delta s_i}$ is thus a non-linear function of the sum of weighted activations $\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j$. In order to express this non-linear relationship, some unit, *h*, must be defined to detect condition **C2**. Recall that the role of unit *i* is to detect condition **C1**. We conclude that at least two different kinds of feature detectors are necessary. The knapsack problem can not be embedded in a network of functionally homogeneous visible units. *Hidden units* are necessary.

In the remainder of this paper, we propose a systematic approach to defining necessary hidden units together with their *best features*. Our approach is applicable to many packing problems including bin packing and multiprocessor scheduling.

Plausible Energy Functions

We have presented an informal argument that hidden units are required by a neural network in order to pack a knapsack. We reinforce this argument by examining two invalid candidate energy functions.

Let

$$E = -A \cdot \sum_{i=1}^{n} s_i p_i + B \cdot \left(\sum_{i=1}^{n} s_i w_i - K \right). \qquad \textbf{E1}$$

This function is clearly inappropriate. Consider the problem instance $Q = \{1\}$, $w_1 = K - \varepsilon$, and $p_1 = \varepsilon$ where $0 < \varepsilon << 1$. For any fixed values of A, B, and K, ε can be selected so that the single element, 1, will not be placed in the knapsack even though it has positive profit and the packing does not overflow. When the algebraic manipulations prescribed by Hopfield and Tank are applied to **E1**, the result is a network in which unit i $(1 \leq i \leq n)$ detects the condition -

$$Bw_i - Ap_i < 0.$$

In this particular network there are no connections. Each unit independently becomes active whenever the weighted profit of its element exceeds the cost regardless of the current contents of the knapsack.

For our second example, let

$$E = -A \cdot \sum_{i=1}^{n} s_i p_i + B \cdot \left(\sum_{i=1}^{n} s_i w_i - K \right)^2. \qquad \textbf{E2}$$

This function would at first glance seem sufficient. It penalizes knapsack overflow and does not reward packings that minimize $\sum_{i=1}^{n} s_i w_i$. It fails in that the energy penalty, ΔE_B, associated with an unused knapsack capacity x is the same as the penalty for an overflow of size x. It is theoretically possible to achieve a configuration, that is, a binary assignment of s_j for all $1 \leq j \leq n$ where $j \neq i$, in which $K - \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j = x$ and $w_i = 2x$ for $x > 0$. The penalty, ΔE_B, resulting from the inclusion of element i in the knapsack is the same as the penalty resulting from the exclusion element i. However, since $p_i > 0$ we have $\frac{\Delta E_A}{\Delta s_i} < 0$. In this configuration, the network will *always* include element i even though the knapsack capacity is exceeded. Varying A and B can not alleviate this invalid bias since the problem lies in the symmetry of the quadratic term E_B.

A Functional Knapsack-Packing Network

We have examined both linear and quadratic forms for the overflow penalty, E_B, and have shown both to be inadequate given the proposed mapping of the problem variable space to the space of activation values. In order to proceed, we acknowledge that an energy function of the activation values of functionally homogeneous units can not have the form of a Hamiltonian and we construct a network, **G1**, of non-neural, binary-state processing units with a non-Hamiltonian energy function.

We define

$$E = E_A + E_B$$

where

$$E_A = -A \sum_{q \in Q'} p_q$$

$$= -A \sum_{i=1}^{n} s_i p_i \ ,$$

$$E_B = \max \left\{ \sum_{q \in Q'} w_q - K, \ 0 \right\}$$

$$= \max \left\{ \sum_{i=1}^{n} s_i w_i - K, \ 0 \right\} , \qquad \textbf{E 3}$$

and A, $B \geq 0$. These terms are intuitively obvious. E_A is minimized by maximizing $\sum_{q \in Q'} p_q$, the profit of the packing. E_B is minimized when knapsack overflow is minimized. Any feasible knapsack packing, Q', for which $\sum_{q \in Q'} w_q \leq K$ will not incur any energy penalty. A simple derivation (see Appendix A) yields

$$\frac{\Delta E}{\Delta s_i} = \frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i}$$

$$\frac{\Delta E_A}{\Delta s_i} = -Ap_i$$

$$\frac{\Delta E_B}{\Delta s_i} = \left[\begin{array}{ll} 0 & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j - w_i \leq K \\ & \\ Bw_i & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j > K \\ & \\ & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \leq K \\ B(w_i - K + \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j) & \\ & \text{and } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j - w_i > K \end{array} \right]$$

From the discrete differential of E, the function of each unit in network **G1** can be determined. Rather than adopting a strict threshold rule of the form -

```
if  $\frac{\Delta E}{\Delta s_i} > 0$  then
        $s_i \leftarrow 0$
else
        $s_i \leftarrow 1$
```

we employ stochastic smoothing of the type used by the units of a Boltzmann machine [Hin84] -

```
if   U(0,1) < $\frac{1}{1 + e^{\left(\frac{-\Delta E}{T \, \Delta s_i}\right)}}$   then
        $s_i \leftarrow 0$
else
        $s_i \leftarrow 1$
```

where **U**(0,1) is a continuous uniform random number. Aarts and Korst [Aar87] have shown stochastic smoothing to be useful when applying Boltzmann Machines to TSP problems. For notational convenience, we will use ΔE to mean $\frac{\Delta E}{\Delta s_i}$ in discussions where there is no ambiguity.

The function of unit *i* in network **G1** is given by -

```
$q_i \leftarrow \sum_{j=1, j \neq i}^{n} s_j w_j$ ;

$\Delta E \leftarrow -Ap_i$ ;

if $q_i > K$ then
    $\Delta E \leftarrow \Delta E + Bw_i$
else if $q_i + w_i > K$ then
    $\Delta E \leftarrow \Delta E + B( w_i + q_i - K )$ ;

if $\mathbf{U}(0,1) < \frac{1}{1 + e^{(-\Delta E / T)}}$ then
    $s_i \leftarrow 0$
else
    $s_i \leftarrow 1$;
```

Unit Function

Binary-State Network **G1**, unit *i* ($1 \leq i \leq n$)

Our next step is to transform **G1** into a *mean-field model* like those of Hopfield and Tank. Let **G2** be a network of analog processing units that is isomorphic to **G1**, our network of discrete units. Each unit *i* of **G2** models the expected value of the activation value of the corresponding unit in **G1**. Thus in network **G2**, unit *i* will be updated according to

$$s_i \leftarrow \frac{1}{1 + e^{(-\Delta E / T)}} .$$

The complete function of unit i in network **G2** is given by -

```
$q_i \leftarrow \sum_{j=1, j \neq i}^{n} s_j w_j$ ;

$\Delta E \leftarrow -Ap_i$ ;

if $q_i > K$ then
     $\Delta E \leftarrow \Delta E + Bwi$
else if $q_i + w_i > K$ then
     $\Delta E \leftarrow \Delta E + B( w_i + q_i - K )$ ;

$s_i \leftarrow \frac{1}{1 + e^{(-\Delta E / T)}}$ ;
```

Unit Function

Analog Network **G2**, unit i $(1 \leq i \leq n)$

Depending on the specific network implementation, **G2** may or may not have the capacity to avoid local minima of its energy function. We postpone a discussion of local minima avoidance to a later section. Network **G2** utilizes asymmetric connections. All connections leaving unit i have weights w_i and every unit has connections to, and from every other unit. Given a digital multi-processor or an array processor, we could easily pack a knapsack by simulating network **G2** with one processor allocated to each unit. It is also possible to design special analog circuitry to evaluate ΔE as a function of $\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j$ in which each activation value, s_j, is modeled by a continuous voltage on [0,+1]. Unfortunately, such a circuit would be decidedly non-neural. Since programmable neural network hardware will reportedly be available in the near future, our objective will be to derive networks of units that model neuron-like behavior, i.e., continuous analog integration and thresholding.

The function of unit i in **G2** can be rewritten as follows -

```
$q_i \leftarrow \sum_{j=1, j \neq i}^{n} s_j w_j$ ;
$\Delta E \leftarrow -Ap_i$ ;
if ($q_i > K$) then
    $\Delta E \leftarrow \Delta E + Bwi$ ;
if ($q_i \leq K$) and ($q_i + w_i > K$) then
    $\Delta E \leftarrow \Delta E + B(w_i + q_i - K)$ ;
$s_i \leftarrow \frac{1}{1 + e^{(-\Delta E / T)}}$ ;
```

Unit Function

Modified Analog Network **G2**, unit i $(1 \leq i \leq n)$

Let us rewrite this function again by replacing implicit binary threshold functions with explicit functions. Let

$$\mathrm{BTHRESH}(x) = \begin{bmatrix} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{bmatrix}$$

The function of unit i in **G2** $(1 \leq i \leq n)$ can be rewritten as

$$q_i \leftarrow \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \; ;$$

$$c1_i \leftarrow \mathrm{BTHRESH}(\; q_i - K \;);$$

$$c2_i \leftarrow \mathrm{BTHRESH}(\; K - q_i + \varepsilon \;);$$

$$c3_i \leftarrow \mathrm{BTHRESH}(\; q_i - K + w_i \;);$$

$$c23_i \leftarrow \mathrm{BTHRESH}(\; Dc2_i + Dc3_i - \frac{3D}{2} \;);$$

$$\Delta E \leftarrow -Ap_i + c1_i \cdot Bw_i + c23_i \cdot B(w_i + q_i - K \;);$$

$$s_i \leftarrow \frac{1}{1 + e^{(-\Delta E / T)}} \; ;$$

Unit Function

Modified Analog Network **G2**, unit i $(1 \leq i \leq n)$

Explicit representation of Binary-Threshold Functions

where $D > 0$ is a constant network parameter (similar to A or B) and $0 < \varepsilon << 1$ is utilized to detect the condition "$q_i = K$". This function deserves a brief explanation. Since $c1_i = 1$ whenever $q_i - K > 0$ and $c1_i = 0$ whenever $q_i - K \leq 0$, by adding $c1_i \cdot Bw_i$ to ΔE we preserve the function of the statement

```
**if** ($q_i > K$) **then**
    $\Delta E \leftarrow \Delta E + Bwi$ ;
```

Since $c23_i = 1$ if, and only if $c2_i = 1$ and $c3_i = 1$, it is clear that $c23_i = 1$ whenever $(q_i \leq K)$ and $(q_i + w_i > K)$ and $c23_i = 0$ otherwise. Multiplication of $B(w_i + q_i - K)$ by $c23_i$ prior to summation with ΔE suffices to increment ΔE by $B(w_i + q_i - K)$ whenever $(q_i \leq K)$ and $(q_i + w_i > K)$. This change preserves the function of the statement

```
**if** $(q_i \leq K)$ **and** $(q_i + w_i > K)$ **then**
    $\Delta E \leftarrow \Delta E + B( w_i + q_i - K )$ ;
```

The energy function associated with network **G2**, does not have a continuous derivative. As $q_i \uparrow K$, unit i strives to increase its level of activation since this produces a decrease in E as dictated by the gradient $\frac{\Delta E_A}{\Delta s_i} = -Ap_i$. At the moment $q_i > K$, the behavior of unit i suddenly changes because the gradient $\frac{\Delta E_B}{\Delta s_i}$ becomes positive. This sudden change in behavior can result in a less efficient packing. For example, consider the problem instance

$$K = 10,$$

$$Q = \{ 1, 2, 3 \}, \; n = 3$$

$$w = (3, 3, 11)$$

$$p = (1, 1, 10)$$

Since the profit of element 3 is relatively large, unit 3 will dominate the activity of the network at moderate temperatures. As the network is cooled and $s_3 \rightarrow 1$ the condition $q_i > K$ will at some time become true. Unfortunately, this will happen at a late point in the cooling schedule and the system may not have sufficient time to settle into a configuration representing a good packing. This situation can of course be remedied by utilizing a longer, more gradual cooling schedule. Increasing the length of the annealing schedule is the standard approach to compensating for coarseness of the energy landscape. However, it is more advantageous to address the underlying problem, the coarseness of the energy landscape, directly.

It is in our interest to smooth the energy landscape defined by **E3** so that, in the case of our example, the system is affected by the "oversize-ness" of element 3 early in relaxation, and the *n*-dimensional trajectory of the system configuration is smooth. In order to smooth the energy landscape of **E3** we utilize the same technique that yielded model **G2** from model **G1** - stochastic smoothing followed by the mean-field transformation. We replace the the function BTHRESH(x) with the function

$$\text{CTHRESH}(x) = \frac{1}{1 + e^{(-x/T)}} .$$

The effect of this replacement is illustrated in figure 2. The result is an efficient, smoothed-energy, analog network which we

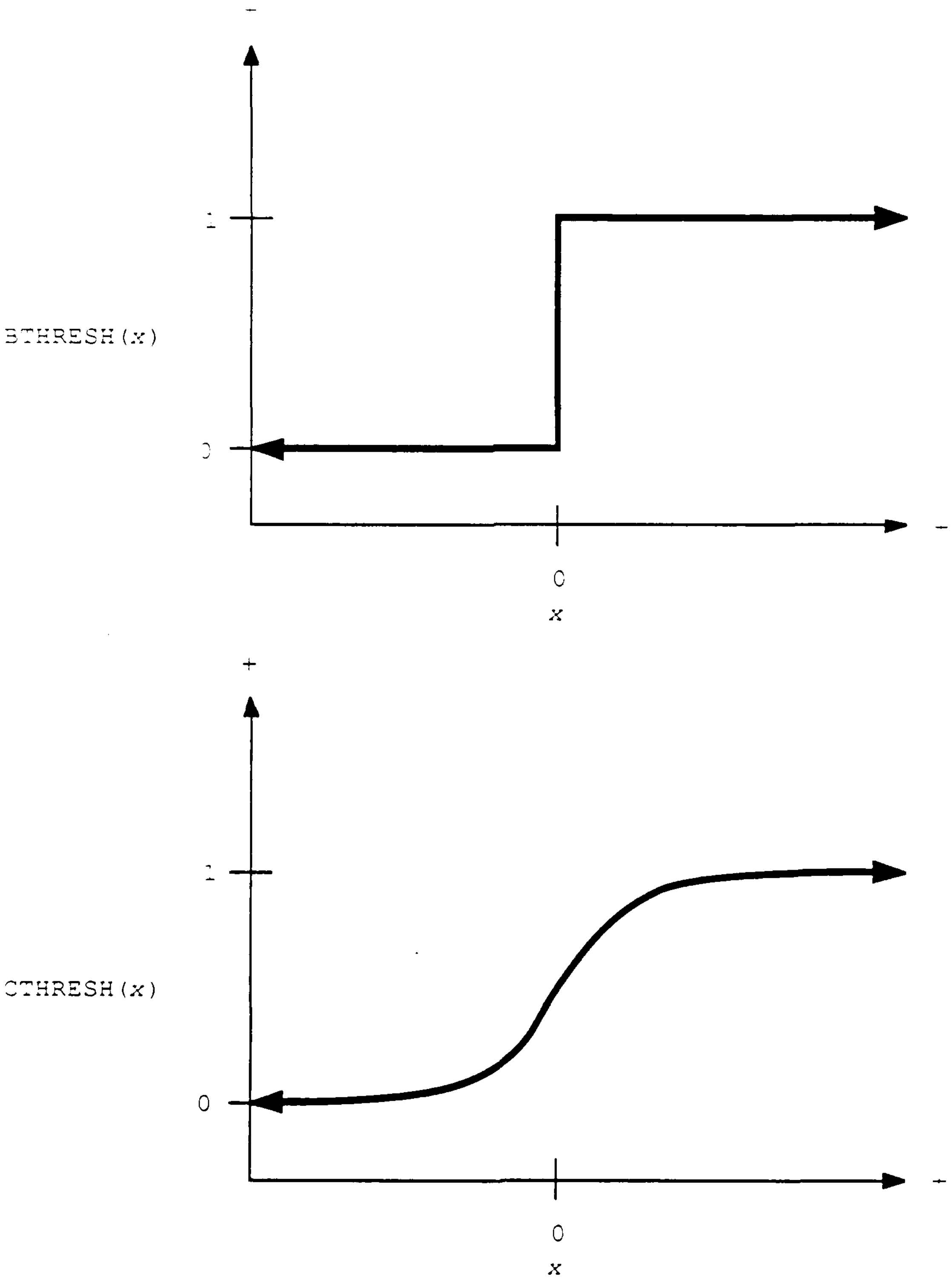


Figure 2

Binary and Graded Threshold Functions

name **G3**. In the low-temperature limit the functions of networks **G2** and **G3** coincide since

$$\lim_{T \downarrow 0} \mathrm{CTHRESH}(x) = \begin{bmatrix} 0 \ \mathit{if} \ x < 0 \\ \\ 1 \ \mathit{if} \ x > 0 \end{bmatrix}$$

$$= \mathrm{BTHRESH}(x) \qquad (x \neq 0) .$$

When the condition $(x = 0)$ must be detected, we add ε to x prior to the application of CTHRESH $(0 < \varepsilon << 1)$.

The function of unit i $(1 \le i \le n)$ in network **G3** is given by

$$q_i \leftarrow \sum_{\substack{j=1 \\ j \ne i}}^{n} s_j w_j \; ;$$

$$c1_i \leftarrow \frac{1}{1 + e^{-\frac{(q_i - K)}{T}}} \; ;$$

$$c2_i \leftarrow \frac{1}{1 + e^{-\frac{(K - q_i + \varepsilon)}{T}}} \; ;$$

$$c3_i \leftarrow \frac{1}{1 + e^{-\frac{(q_i + w_i - K)}{T}}} \; ;$$

$$c23_i \leftarrow \frac{1}{1 + e^{-\frac{(Dc2_i + Dc3_i - \frac{3D}{2})}{T}}} \; ;$$

$$\Delta E \leftarrow Ap_i - c1_i \cdot B w_i - c23_i \cdot B \, (\, w_i + q_i - K \,) \, ;$$

$$s_i \leftarrow \frac{1}{1 + e^{-\Delta E / T}} \; ;$$

Unit Function

Smoothed-Energy Analog Network **G3**, unit i $(1 \le i \le n)$

We are now in a position to make an interesting observation. Since the values $c1_i$, $c2_i$, $c3_i$, and $c23_i$ are computed continuously, if we prevent these values from changing too rapidly, then $c1_i$, $c2_i$, $c3_i$, and $c23_i$ can be computed *in parallel*. We simply replace the entire unit with a network of 5 simpler units as shown in figure 3. We refer to these units as *c1-*, *c2-*, *c3-*, *c23-*, and *s*-units. The functions of these units, described in algorithmic form, is shown in figure 4. We refer to the resulting network as **G4**.

The astute reader will notice that we have introduced neuron-like behavior to each unit. We can now interpret the functions of all units as *feature detectors* as shown in figure 5. The only anomaly of network **G4** is in the function of the *s*-unit which must compute a weighted product of the incoming activation signals from other *s*-units and its *c23*-unit. This special case is represented by a *higher-order interaction* or *conjunctive synapse*. The activation value of the *c23*-unit must moderate the transmission of the activation values of all external *s*-units to the *s*-unit in its cluster. It is well known that the function of conjunctive synapses can be approximated, in the low-temperature limit, by conjunctive units. Each conjunctive synapse is simply replaced by a separate unit that detects the conjunction of the *c23*-unit and the *s*-unit as shown in figure 6. We note that

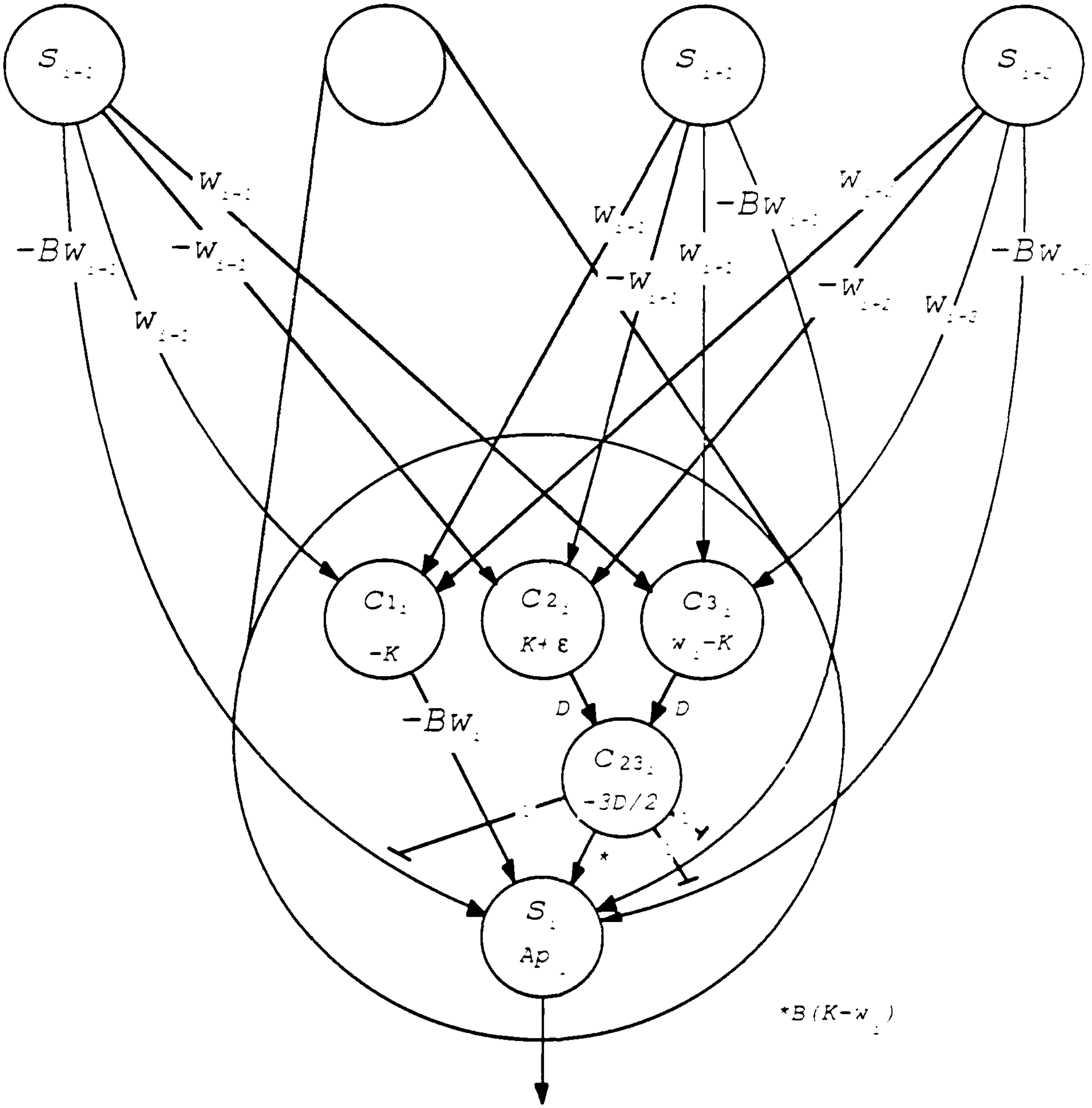
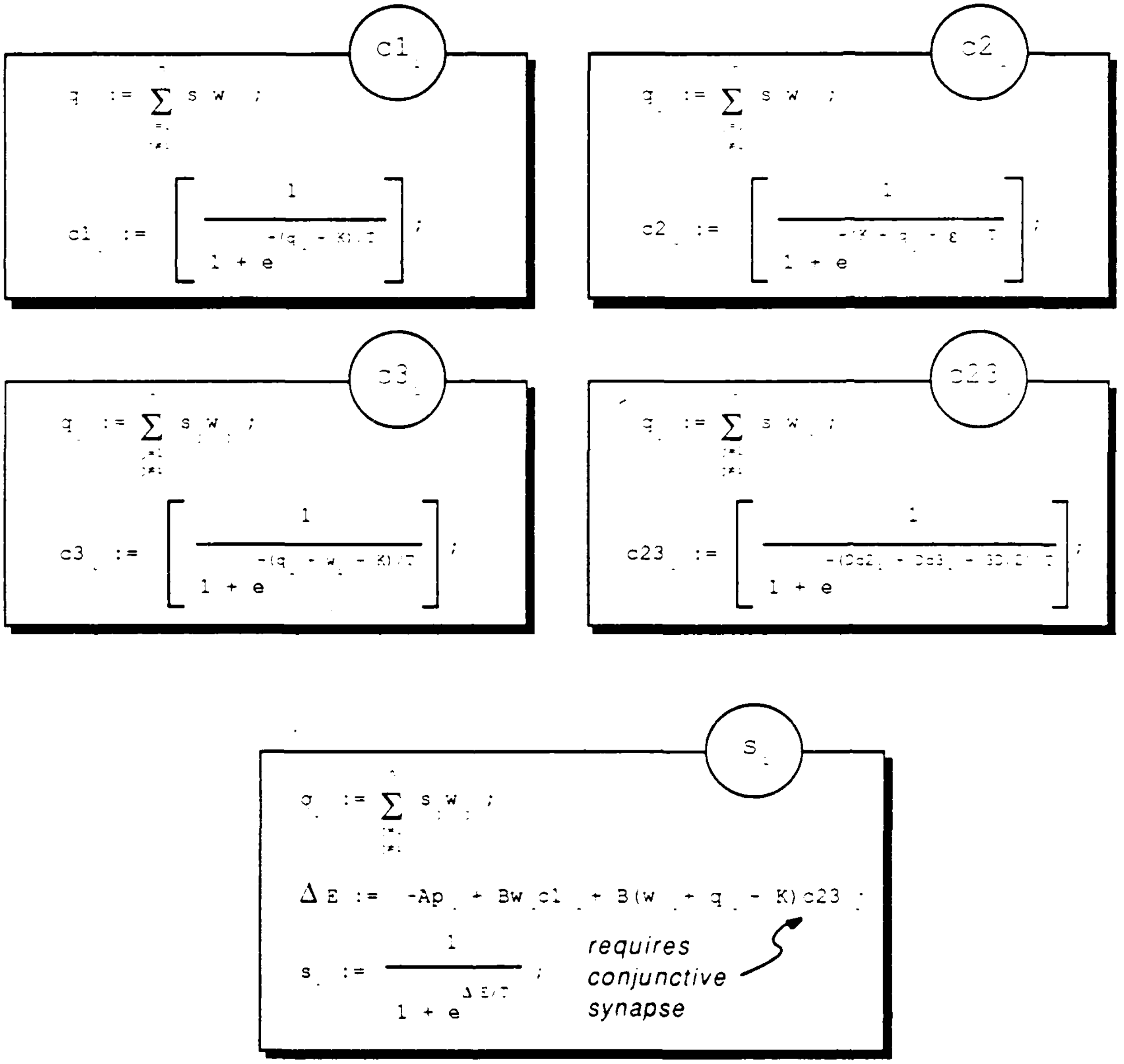


Figure 3

Analog Knapsack-Packing Network G4

with conjunctive synapses



$$q_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j ;$$

$$c1_i := \left[\frac{1}{1 + e^{-(q_i - K)/T}}\right];$$

$$q_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j ;$$

$$c2_i := \left[\frac{1}{1 + e^{-(K - q_i - \varepsilon)/T}}\right];$$

$$q_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j ;$$

$$c3_i := \left[\frac{1}{1 + e^{-(q_i + w_i - K)/T}}\right];$$

$$q_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j ;$$

$$c23_i := \left[\frac{1}{1 + e^{-(Dc2_i + Dc3_i - 3D/2)/T}}\right];$$

$$q_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j ;$$

$$\Delta E := -Ap_i + Bw_i c1_i + B(w_i + q_i - K)c23_i ;$$

$$s_i := \frac{1}{1 + e^{\Delta E/T}} ;$$

Figure 4

Parallel Decomposition of Non-Neural Knapsack Unit (G3)

into 4 Hidden and 1 Visible Unit (G4)

Hidden Units :

$c1_i$ Detects when the knapsack overflows given that element i is not in the knapsack.

$c2_i$ Detects the absence of overflow in the knapsack given that element i is not in the knapsack.

$c3_i$ Detects when the knapsack overflows given that element i is in the knapsack.

$c23_i$ Detects the conjunction of the conditions detected by $c2_i$ and $c3_i$, that is, when element i produces an overflowing knapsack.

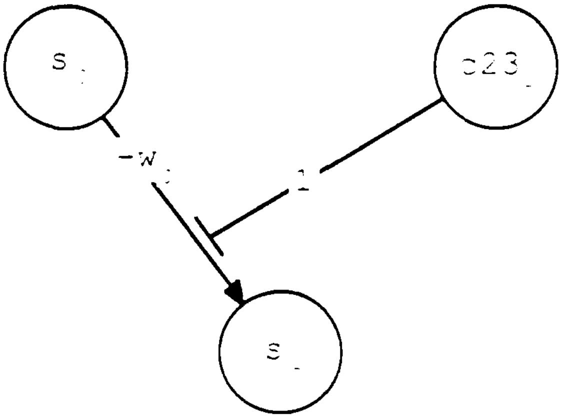
Visible Units :

s_i Detects the condition whereby the benefit of including element i in the knapsack exceeds the cost.

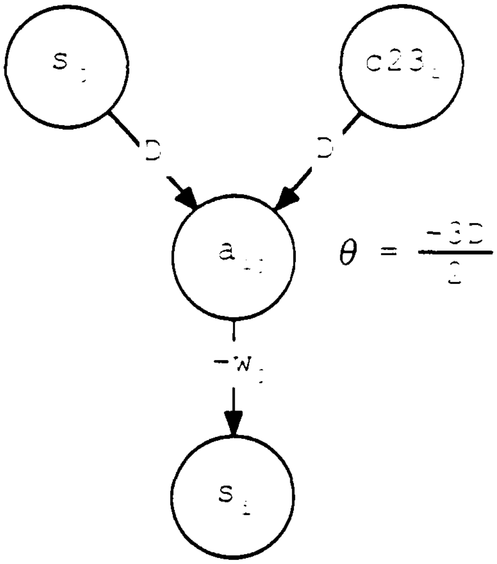
Figure 5

Knapsack Units as Feature Detectors

Conjunctive synapses of the form :



can be replaced, in the low-temperature limit, by conjunctive units of the form :



where D is a constant used to control the "decisiveness" of the conjunction. D > 0.

Figure 6

Reduction of Synaptic Order

$$B\ c23_i\ q_i \;=\; B\ c23_i \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j$$

$$= B \sum_{\substack{j=1\\ j\neq i}}^{n} c23_i\, s_j w_j\ .$$

We define

$$a_{ij} = \frac{1}{1 + e^{\left(-\frac{Dc23_i + Ds_j - \frac{3D}{2}}{T}\right)}}$$

so that

$$\lim_{T\downarrow 0}\ a_{ij} = \lim_{T\downarrow 0}\ c23_i\, s_j$$

and

$$\lim_{T\downarrow 0}\ B\, c23_i\ q_i \;=\; \lim_{T\downarrow 0}\ B \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij} w_j\ .$$

After completing this reduction of synaptic order, we rewrite the function of the *s*-unit by

$$p_i := \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} w_j ;$$

$$\Delta E := -Ap_i - Bw_i c1_i - B(w_i - K)c23_i - Bq_i ;$$

$$s_i := \frac{1}{1 + e^{\Delta E/T}} ;$$

The function of unit a_{ij} $(1 \leq i, j \leq n, i \neq j)$ is

$$a_{ij} := \left[\frac{1}{1 + e^{-(Dc23_i + Ds_j - 3D/2)/T}} \right] ;$$

This order reduction increases the total number of units in the network from $O(n)$ to $O(n^2)$ but results in a complete network, **G5**, of neuron-like processing units connected with simple synapses. An example of network **G5** for a 3-element knapsack problem (i.e. $|Q|=3$) is shown in figure 7.

Summary of the Derivation Method

As we did for the case of the set partition network, let us summarize the steps that were used to arrive at the knapsack-packing network.

1 A network representation is selected. n, the dimensionality of the network state space (alternatively, the number of units in the network architecture) is defined. A transformation, M:$\mathbf{V}\rightarrow\mathbf{S}'$, from $\mathbf{V}$, the space of problem variable configurations, to $\mathbf{S}' = \{0,1\}^n$, the corners of the n-dimensional unit hypercube, is defined.

2 An inverse transformation, M^{-1}:$\mathbf{S}\rightarrow\mathbf{V}$, is defined. Each network configuration in the volume of the n-dimensional hypercube is mapped by M^{-1} to a configuration of the variables of the problem space.

3 A discrete energy function, $E:\mathbf{S}'\rightarrow\Re$, is defined on $\mathbf{S}'$ so that $M^{-1}(\mathbf{s}_{\text{min}})$ is a minimum of the problem objective, f, whenever $\mathbf{s}_{\text{min}}$, is a minimum of E, $\mathbf{s}_{\text{min}}\in\mathbf{S}$. E need not take the form of a 0-1 Hamiltonian.

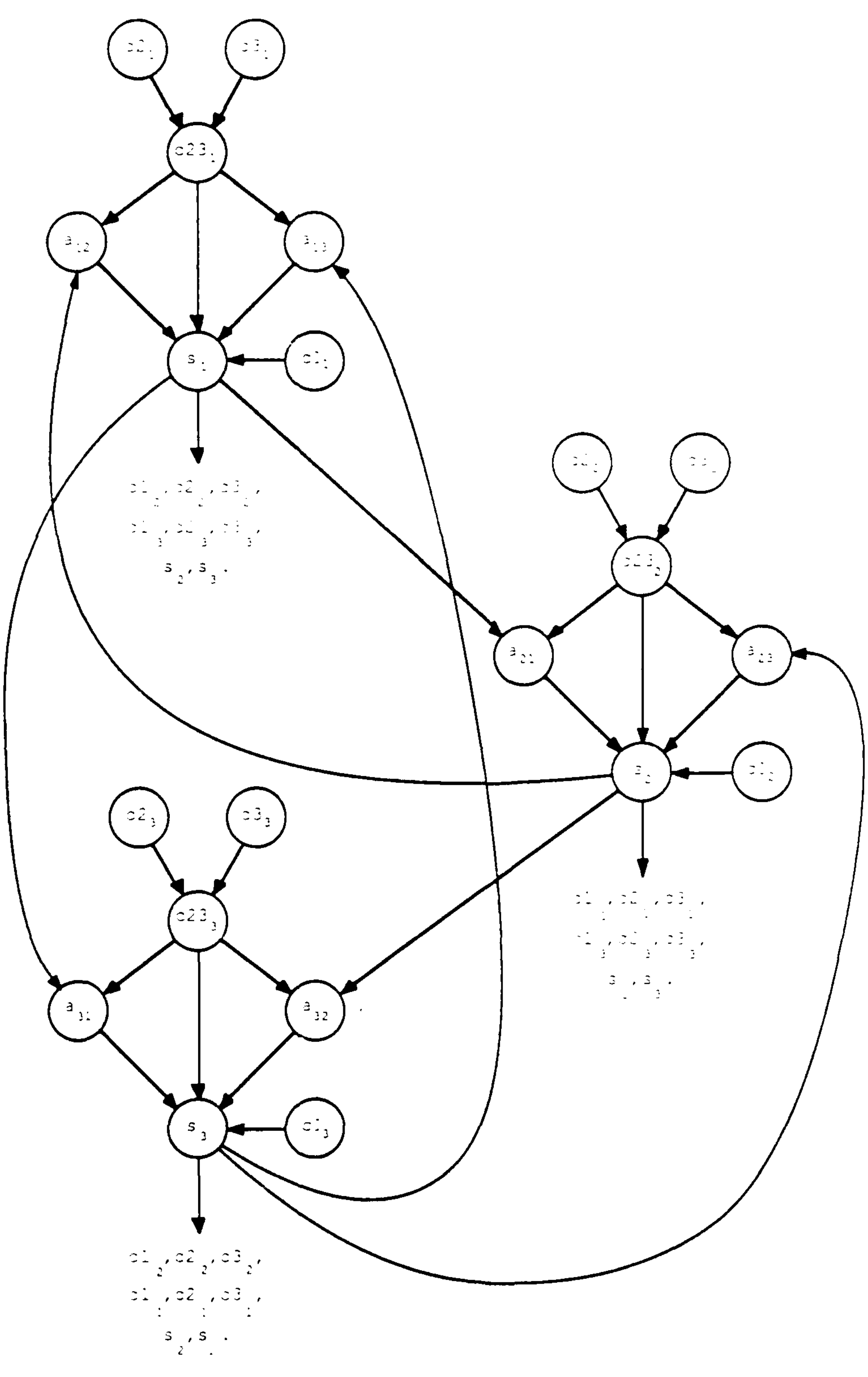


Figure 7

3-Element Analog Knapsack-Packing Network

4 The discrete partial differential, $\frac{\Delta E}{\Delta s_i}$, is derived giving the gradient of E at the corners of the hypercube.

5 A network of non-neural, binary-state processing units is constructed (e.g. **G1**). Stochastic smoothing is used to avoid local minima.

6 The mean-field transformation is applied to the binary-state model of step 5 yielding a network of non-neural, analog processing units. This transformation provides an interpolation of $\frac{\Delta E}{\Delta s_i}$ in the interior of the hypercube, **S**.

7 Implicit binary-threshold functions are identified and rewritten explicitly via the BTHRESH function. Auxiliary binary variables (e.g. *c23*) are assigned to represent boolean combinations (***and***, ***or***, etc.) of simple conditions.

8 The BTHRESH function is replaced with the sigmoid CTHRESH function. This smooths the energy landscape.

9 The entire non-neural analog unit is replaced with a collection of hidden units together with a single visible unit. A hidden unit is introduced for each condition, that is, each application of CTHRESH.

10 The network is relaxed using one of a variety of unit update rules.

11 After relaxation, M^{-1} is applied to the final network configuration yielding a minimum of f, the objective function of the original optimization problem.

This derivation technique differs from the standard technique in two important respects. The initial proposed energy function E need not take the form of a Hamiltonian and need not even be continuously differentiable on the interior of the hypercube. Consequently, we will not have neuron-like unit behavior, i.e., summation and thresholding. Neuron-like behavior must be re-introduced by steps 6-9. More importantly, explicit recognition of implicit binary thresholds that are present in $\frac{\Delta E}{\Delta s_i}$ allows us to eventually replace these functions with hidden units. The constructive method, that is, the sequence of function replacements, simplifies the conceptual "leap" that must be taken when reducing non-trivial discrete optimization problems to neural-network algorithms. The method is summarized in figure 8.

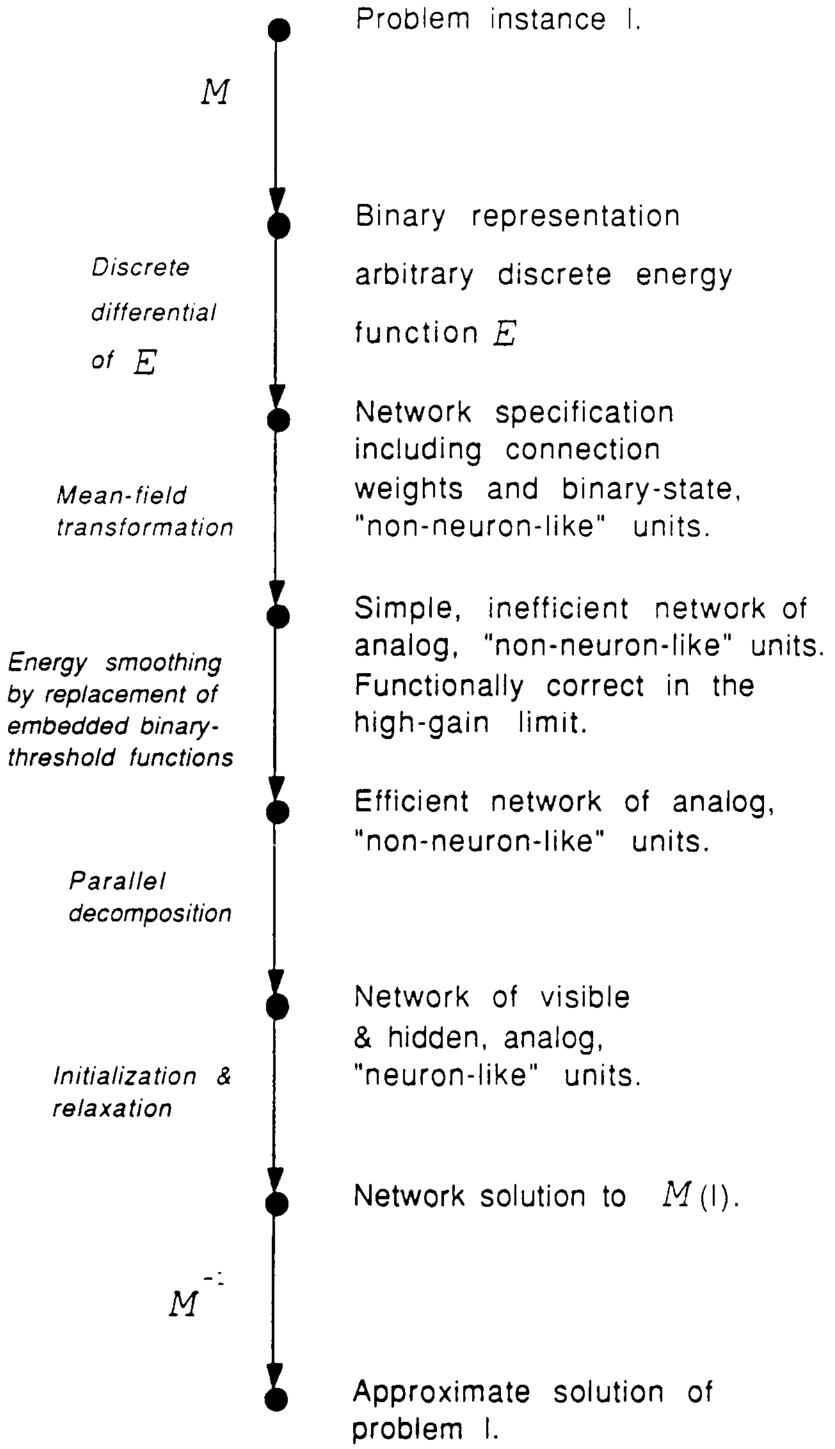


Figure 8

Proposed Thermodynamic Neural Network Derivation Procedure

Global Timing Considerations

A number of issues must be considered when implementing neural networks. We must specify network and operational parameters and an annealing schedule. In continuous-time systems the response curves of the components comprising the connections and units must be specified. In our implementations, time is discrete so that we must specify the scheduling of unit updates. In this section we address the last of these topics - global timing.

It is well known that Hopfield and Tank's models prescribe deterministic trajectories through configuration space during network relaxation. As such, they are subject to entrapment in local minima of E. Hopfield and Tank modeled the non-linear response of neuron-like units with non-linear amplifiers. By increasing the *gain*, λ (note: $\lambda \sim \frac{1}{T}$), during relaxation deeper energy minima were located. In a noise-free simulation, varying the gain is inconsequential and the final configuration of the network depends solely on the initial configuration and its associated basin of attraction. During simulation, numerical noise may introduce counter-gradient transitions and could account for Hopfield and Tank's slightly improved results. Unfortunately, numerical noise is not easily characterized.

Boltzmann machines can exit local minima because they possess a well-defined *generation mechanism* to introduce counter-gradient transitions or "uphill jumps" in E. Unfortunately, the variance in the binary activation states of Boltzmann units is relatively large and results in unacceptably long relaxation time requirements. In addition to the non-deterministic state transitions of Boltzmann units, another mechanism for generating counter-gradient transitions is present - asynchronism. Randomly probing units for update is equivalent to simulating random propagation delays along connections. In the Boltzmann machine, the mean propagation delay is *n update periods* where *n* is the number of free-running units (*unclamped units* in the terminology of Ackley, Hinton, and Sejnowski [Hin84]).

In our implementation, we combine both the continuous state-space of Hopfield networks together with the asynchronism of the Boltzmann machine. This is accomplished by using deterministic unit update rules (those of networks **G2** and **G3**) and by randomly scheduling each unit for update based on a discrete uniform (1,*n*) distribution. At update, each unit's next state is computed and immediately adopted. This combination yields rapid relaxation with a well-defined, controllable source of counter-gradient transitions. Random propagation delays induce gaussian *internal noise* into the activation value of each unit. The variance of the noise is a function of the distribution of propagation delays as well as the entropy of the system.

There are several interesting interactions between internal noise and the annealing process. We have stated that internal noise is partially dependent on the system entropy. As the network stabilizes at a minima, the motion of its global state though n-dimensional space decreases and activation levels of units are modified by successively smaller amounts. As a result, the effect of propagation delays becomes less pronounced. For example, let us assume that the signal, s_i, from unit i to unit j is delayed along connection (i,j). If unit i has stabilized then s_i will not have changed since its previous value was received by unit j so that unit j is receives the current value of s_i. In this case, the signal delay has no effect. As propagation delays effectively disappear, the variance of internal noise decreases. Let us briefly examine the dynamics of a simple annealed system [Kir83]. As $T \downarrow 0$, fewer random perturbations are accepted and the progress of the global state toward a minima degenerates. Although it is theoretically possible for the system to make a large counter-gradient transition at low temperature, the probability of such a transition decreases exponentially. We can improve the productivity of the annealing algorithm by decreasing the average "height", ΔE, of proposed counter-gradient transitions as the system entropy decreases. This is precisely the effect that is achieved by propagation delay-induced internal noise since its variance decreases as the network configuration approaches a local minimum.

The sigmoid transfer function -

$$g(x) = \frac{1}{1 + e^{-x/T}}$$

has stable points at 0, and 1. g has a meta-stable point at 0.5. When $\frac{\Delta E}{\Delta s_i} > 0.5$ and $\frac{\Delta E}{\Delta s_i} < 0.5$, the state, s_i, of unit i will have a tendency to move toward 1 and 0 respectively. If s_i is close to 0.5 (and unit i is "indecisive" or "teetering") the effect of internal noise on s_i is more pronounced. In a meta-stable state, even a small amount of noise may "tip the scale" and start the the network on a trajectory to a different final configuration. As the network stabilizes and its configuration reaches a corner of the hypercube, not only does the variance of internal noise decline, but the probability that a noise surge will result in the transition of unit i from $s_i < 0.5$ to $s_i > 0.5$ (or the converse) decreases since the difference between s_i and 0.5 increases as the network settles.

Simulation

A number of simulations were performed on networks **G3** and **G4**. In all simulations an asynchronous, sequential update rule like that of the Boltzmann machine was used. Prior to relaxation the activation values of all units were initialized to uniform (0.4,0.6) random numbers. Simple fixed-length annealing schedules of 10 or 20 fixed-temperature "chains" were used (table 1).

Number of Update Cycles	Temperature	Number of Update Cycles	Temperature
2	20.0000	32	1.2302
2	15.1329	42	0.9308
3	11.4503	56	0.7043
4	8.6638	73	0.5329
6	6.5555	99	0.4032
8	4.9602	131	0.3051
10	3.7531	173	0.2309
14	2.8398	229	0.1747
18	2.1487	302	0.1322
24	1.6258	400	0.1000

Table 1

20 Step Exponential Annealing Schedule

At each temperature, a fixed number of update cycles were executed. For each update cycle, *n* random unit probes were made. Each time a unit was probed, its state was updated according to its prescribed local function. The constant ε, used to bias the *c23-* units, was arbitrarily set to 0.01 and the threshold of M^{-1}, χ , was set to 0.5.

The network parameter *D* is used to control the "decisiveness" of the conjunctive *c23-* units. Since

$$e^{-\frac{(Dc2_i + Dc3_i - \frac{3D}{2})}{T}} = e^{-\frac{(c2_i + c3_i - \frac{3}{2})}{TD^{-1}}}$$

D^{-1} can be interpreted as the ratio of the temperature of the *c23-* units to the temperature of the other units In order for the *c23-* units to run at roughly the same temperature as all other units, we defined *D* as a function of *A* and *B*, specifically, *D* is set to the average absolute value of all connection weights and biases excluding both the connections directed into the *c23-* units and the biases of the *c23-* units.

The remaining network parameters, *A* and *B*, are related. *A* and *B* are coefficients of energy terms that represent conflicting ground states. By increasing *A* relative to *B* we emphasize profitable packings but increase the likelihood of exceeding the knapsack capacity. By increasing *B* relative to *A*, we emphasize valid packings (i.e. packings for which $\sum_{q \in Q'} w_q \leq K$) and give less importance to the profit, $\sum_{q \in Q'} p_q$, of the packing. In our simulations *A* was fixed at 6.0 and different values of *B* were tested.

Problem instances were randomly generated as a function of *K*. Specifically, $n \sim \mathbf{U}(K-5, K+5)$, $w_i \sim \mathbf{U}(1, K)$, and $p_i \sim \mathbf{U}(1, K)$ for $1 \leq i \leq n$. Several values of *K* (10, 20, 35, and 80) were tested. For each combination of network parameters, and for each problem instance, 10 simulations were performed. After a network was constructed for each problem instance, the network was relaxed using the 10 or 20 step annealing schedule. The resulting approximate solutions were compared with those produced by the standard "greedy" algorithm - a first-fit-decreasing packing by value/density ratios $\frac{p_i}{w_i}$. The performance ratios, $\frac{\text{NETWORK(I)}}{\text{GREEDY(I)}}$, are shown in table 2. On the average, the knapsack-packing network performed slightly better than the greedy algorithm.

B	% Valid Packings	$\frac{\text{NETWORK(I)}}{\text{GREEDY(I)}}$
4	60	0.9660
5	80	1.0582
6	100	1.0718
8	90	1.0069
9	95	1.0578
10	100	1.0593
12	100	1.0701
14	100	1.0380
20	100	1.0822

Table 2

Effect of Varying Network Parameter *B*

Network G4, $K = 20$, 20 step annealing schedule, $A = 6.0$

Out of curiosity, 20 simulations of network **G2** were executed with ~200 elements. This would correspond to a **G4** network of 1000 units. No effort was made to tune the network parameters, *A* and *B*, to improve the results of this particular simulation ($A = 6$ and $B = 14$ were used). Nevertheless, a performance ratio of 0.9721 was achieved. This result suggests that degradation due to increased frustration from scaling may not be as serious a problem as we expected

Conclusion

We have seen that certain optimization problems can not be embedded in neural networks that utilize simple mappings of problem variables to *n*-dimensional spaces of activation values. For these problems, it is necessary to define hidden units that detect different types of features. Although we have not discounted the possibility of discovering hidden units and their best features by sheer cleverness, a more systematic method is warranted. The method that we have presented is suitable for objective functions that are not expressible as 0-1 Hamiltonians, or those that are not continuously differentiable over the interior of the unit hypercube.

A range of computational networks that with -

1 Complex non-neural, binary-state units (**G1**),

2 Complex non-neural, continuous-state units (**G2**,**G3**),

3 Simple, neuron-like, continuous-state units with conjunctive synapses (**G4**), and

4 Simple, neuron-like, continuous-state units with first-order synapses (**G5**)

have been presented for the integer knapsack packing problem. Preliminary simulation results are promising, especially in light of the ease with which we were able to find viable network parameters. Simulation results indicate that these networks appear to scale reasonably well. In the course of additional research we have found the method to be applicable to bin-packing, multiprocessor scheduling, and job-sequencing problems as well. We hope to simulate networks for these problems in the near future.

Appendix A

We derive $\frac{\Delta E}{\Delta s_i}$ for $1 \leq i \leq n$. We are given

$$E = E_A + E_B$$

where

$$E_A = -A \sum_{i=1}^{n} s_i p_i$$

and

$$E_B = \max \left\{ \sum_{i=1}^{n} s_i w_i - K, \ 0 \right\}.$$

$$\frac{\Delta E}{\Delta s_i} = \frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i}$$

$$E_A \Big] s_i = 0 = -A \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j p_j$$

$$E_A \Big] s_i = 1 = -A \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j p_j + p_i \right)$$

$$\frac{\Delta E_A}{\Delta s_i} = E_A \Big] s_i = 1 - E_A \Big] s_i = 0$$

$$= -Ap_i \tag{1}$$

$$E_B\Big]_{s_i = 0} = \left[\begin{array}{ll} 0 & \text{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \leq K \\ B\left(\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j - K\right) & \text{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j > K \end{array}\right]$$

$$E_B\Big]_{s_i = 1} = \left[\begin{array}{ll} 0 & \text{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i \leq K \\ B\left(\sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i - K\right) & \text{if } \sum_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i > K \end{array}\right]$$

When combining $E_B\Big]_{s_i = 0}$ and $E_B\Big]_{s_i = 1}$ four conditions must be considered -

Condition 1

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j \leq K \quad \wedge \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i \leq K$$

Since $w_i \geq 0$ for $1 \leq i \leq n$,

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i \leq K \quad \rightarrow \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j \leq K$$

so that

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i \leq K \quad \rightarrow \quad \left(E_B \right]_{s_i = 1} - E_B \left. \right]_{s_i = 0} \Big) = 0 . \qquad (2)$$

Condition 2

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j \leq K \quad \wedge \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i > K$$

$$\rightarrow \quad \left(E_B \right]_{s_i = 1} - E_B \left. \right]_{s_i = 0} \Big) = B \left(\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i - K \right) . \qquad (3)$$

Condition 3

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j > K \quad \wedge \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i \leq K$$

Since $w_i \geq 0$ for $1 \leq i \leq n$, condition 3 vacuously false.

Condition 4

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j > K \quad \wedge \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j + w_i > K$$

Again, since $w_i \geq 0$ for $1 \leq i \leq n$,

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j > K \quad \rightarrow \quad \sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j - w_i > K$$

so that

$$\sum_{\substack{j=1\\ j\neq i}}^{n} s_j w_j > K \quad \rightarrow \quad \left(E_B \right]_{s_i=1} - \left. E_B \right]_{s_i=0} \Big) = Bw_i . \qquad (4)$$

Combining (2), (3), and (4) we have

$$\frac{\Delta E_B}{\Delta s_i} = E_B\Big]_{s_i=1} - E_B\Big]_{s_i=0} \tag{5}$$

$$= \left[\begin{array}{ll} 0 & \text{if } \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j + w_i \leq K \\[2em] B w_i & \text{if } \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j > K \\[2em] B\left(w_i - K + \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j\right) & \text{if } \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j \leq K \\ & \text{and } \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} s_j w_j + w_i > K \end{array}\right]$$

The summation of formulas (1) and (5) gives the desired result, i.e., a complete formula for $\frac{\Delta E}{\Delta s_i}$ -

$$\frac{\Delta E}{\Delta s_i} = \frac{\Delta E_A}{\Delta s_i} + \frac{\Delta E_B}{\Delta s_i}$$

$$\frac{\Delta E_A}{\Delta s_i} = -Ap_i$$

$$\frac{\Delta E_B}{\Delta s_i} = \left[\begin{array}{ll} 0 & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i \leq K \\ & \\ Bw_i & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j > K \\ & \\ B\left(w_i - K + \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j\right) & \text{if } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j \leq K \\ & \text{and } \sum\limits_{\substack{j=1 \\ j \neq i}}^{n} s_j w_j + w_i > K \end{array}\right]$$

This completes our derivation.

References

Aar87 Aarts, E.H.L., and Korst, J.H.M., Boltzmann Machines and their Applications, *Proc. conf. on Parallel Architectures and Languages Europe*, Eindhoven, June, 1987, Springer Lecture Notes, Springer-Verlag, Berlin, 1987.

Dah88 Dahl, E.D., Neural Network Algorithm for an *NP*-complete problem: Map and Graph Coloring, *IEEE ICNN conf. proc.*, Vol. 3, pp 113-120, June, 1988

Gar79 Garey, M., and Johnson, D., Computers and Intractability; A Guide to the theory of *NP*-completeness, Freeman, 1979.

Hin84 Hinton, G.E., Sejnowski, T.J., Ackley, D.H., Boltzmann Machines: Constraint Satisfaction Networks that Learn, *Tech. Rep. No. CMU-CS-84-119*, Carnegie-Melon University, Pittsburgh, PA, May, 1984.

Hop85 Hopfield, J.J., and Tank, D.W., "Neural" Computation of Decisions in Optimization Problems, *Biological Cybernetics*, Vol. 52, pp. 141-152, Springer-Verlag, 1985.

Kir83 Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., Optimization by Simulated Physical Annealing, *Science*, Vol. 220, No. 4593, pp. 671-680, May 1983.

Ram88 Ramanujam, J., Sadayappan, P., Optimization by Neural Networks, *IEEE ICNN conf. proc.*, Vol. 2, pp 325-332, June, 1988